Decomposition of Markov Decision Processes Using Directed Graphs

Pierre Laroche, François Charpillet, René Schott

Decomposition of Markov Decision Processes Using Directed Graphs

Pierre Laroche, François Charpillet and René Schott

LORIA INRIA-Lorraine - Campus scientifique, BP 239 - 54506 Vandœuvre-lès-Nancy
Fax: +33 3.83.41.30.79 - Email : laroche,charp,schott{@loria.fr}

1 Introduction

Decomposing an MDP consists in (1) partitioning the state space into regions; (2) solving each region independently; (3) combining the local solutions to generate a global one. [Dean and Lin1995] give a general framework to decompose MDPs, but the process they propose is quite complex. Other researches aim to use *macro-actions* to efficiently solve MDPs [Parr1998]. In our approach, domain characteristics are used to accurately define how region communicates with each others, which is very important to ensure good quality of the approximate policies obtained. We apply it to mobile robotics planning in indoor environments.

2 Markov Decision Processes

An MDP can be formally defined as a tuple $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ where $\mathcal{S}$ is a finite set of states of the environment, $\mathcal{A}$ is a finite set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state transition function, and $R : S \rightarrow \mathbb{R}$ is the reward function. An optimal policy π defines an optimal action for each state of $\mathcal{S}$. The *Policy Iteration* algorithm [Howard1960] iteratively maximizes the value of each state, which is formulated as follows: $V_\pi(s) = R(s) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') V_\pi(s')$ (1).

3 Our Method

To compute fast and efficient policies, we represent the environment using a graph. The partition is given by a hand-labelled description of the environment, casting states into corridor/office or intersection states. Each intersection becomes a vertex, and corridors or offices between two intersections become arcs. Instead of using the classical value function (equ. 1), arcs are valuated using two criteria (distance to the goal and risk of collision), using the following function:

$$V_g(v_n) = V_g(v_{n-1}) + \sum_{i \in v_n \rightarrow v_{n-1}} \gamma \, r_{v_n \rightarrow v_{n-1}} \, (\texttt{VMAX} - V_g(i))$$

Here vertex v_n is valuated using the value of its best successor v_{n-1} on the way to the goal (according to a classical Dijkstra algorithm), at which is added the

cost of reaching it. This cost is computed as a fraction of the value which can be added to the value of v_{n-1}. This fraction depends on the risk of collision r, which is computed using the transition function, according to width of corridors and obstacles to avoid. The value `VMAX` is the biggest value a state can have. In figure 1 is shown the graph obtained using our method for a small environment. Each arc (each corridor) becomes a sub-MDP, whose goals are the adjacent intersections. To represent the attraction of each region over its neighbours, the arc values are used to define the values of the sub-MDPs goals. For instance, the sub-MDP representing region {`Vertex4` $\leftrightarrow$ `Vertex5`} has two goals, one with a value of 84.22 (56.95+27,27), the other with a value of 91.32. As a consequence, vertex 4 is more attractive, which corresponds to the optimal policy.

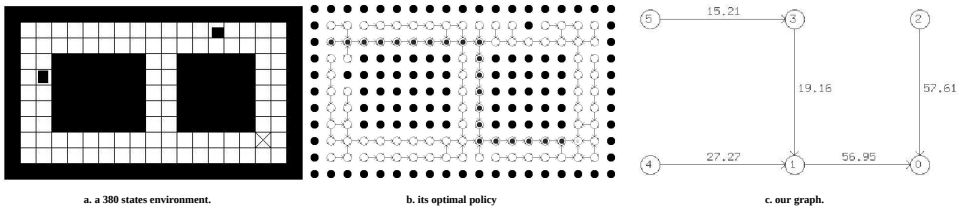

Fig. 1. An environment, its optimal policy and the graph built to approximate it.

To test our approach, we used several big environments (approx. 4000 states). The average quality obtained is 95% of the optimal solution, whereas a deterministic policy (optimality criteria is just the distance to the goal) is only 65% of the optimal. The worst result is 84% of the optimal, whereas the deterministic algorithm gives a worst result of only 26%. The computing time is reduced from 1510 seconds using *Policy Iteration* to 29 seconds using our approach.

4 Concluding Remarks

In our approach, characteristics of the domain are used to define how regions communicate which each others. It can be applied to MDPs which can be represented as a graph, and whose policy depends on a set of features of the environment. High-quality policies are obtained in a reduced computing time.

References

[Dean and Lin1995] Thomas Dean and Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *Proc. of IJCAI*, 1995.

[Howard1960] Ronald A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, Massachussets, 1960.

[Parr1998] Ronald Parr. Flexible decomposition algorithms for weakly coupled markov decision problems. In *Proc. of UAI*, 1998.

[Puterman1994] Martin L. Puterman. *Markov Decision Processes.* Wiley & Sons, 1994.